UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

MATERIA: SISTEMAS DE GESTIÓN DE BASES DE DATOS
PRIMER LABORATORIO “ADMINISTRACIÓN BÁSICA DE UN SGBDR”

NIVEL: 3

CICLO LECTIVO: 2024 – 2DO. CUATRIMESTRE

DOCENTES:
PROFESOR: I.S.I. ANDRÉS PABLO FANTÍN
J.T.P.: I.S.I. JUAN CARLOS FERNÁNDEZ
AUXILIAR ADSCRIPTA: LUCIANA CAMPESTRINI

Grupo n.º2:
-Velazco Gez Schegtel, Juan Ignacio

Para realizar el laboratorio se utilizará la base de datos “employees”. La misma la descargué utilizando la guía que se encuentra en la documentación oficial de MySQL, en mi máquina local en la carpeta compartida con la máquina virtual de Multipass:

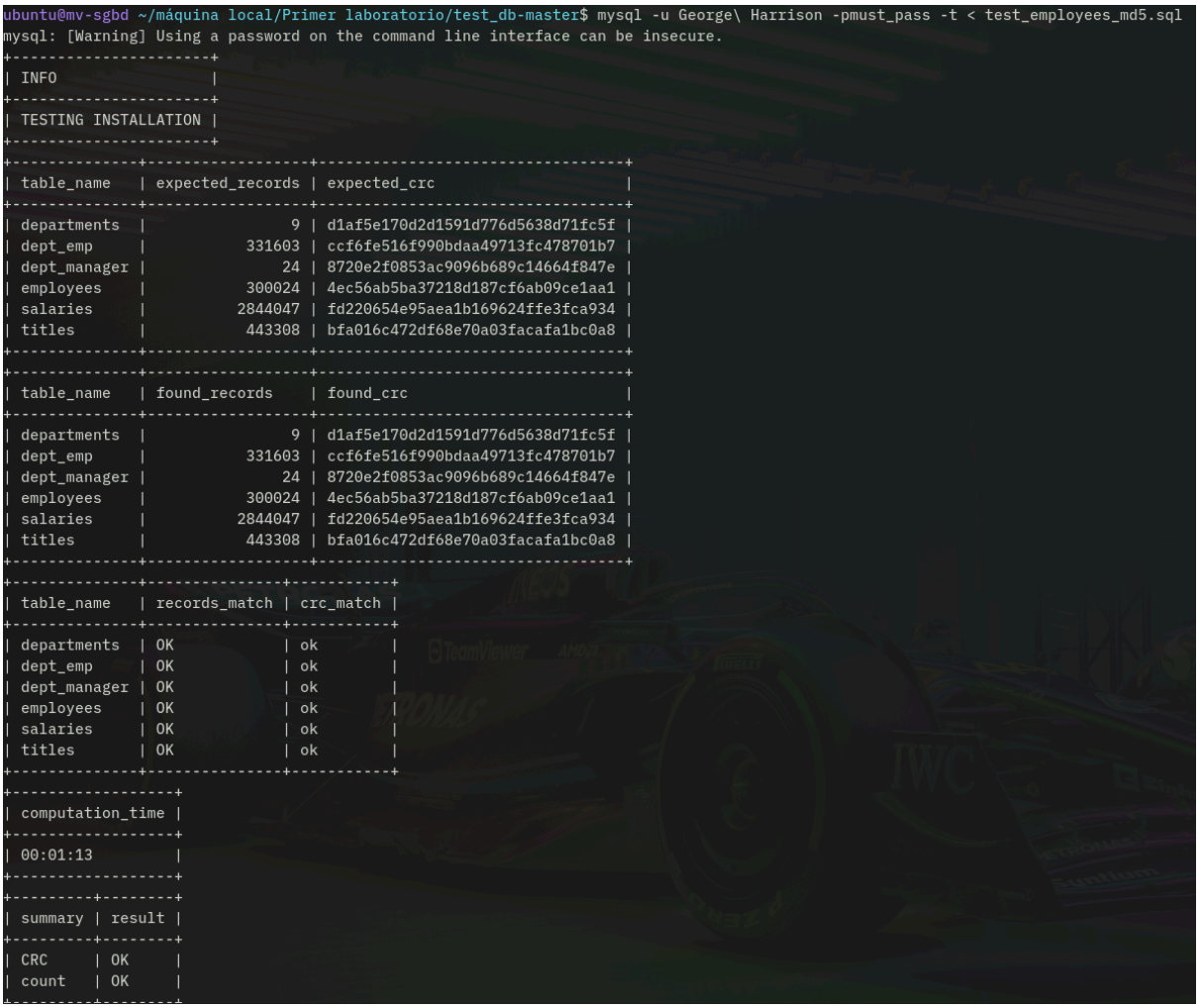
```
~/OneDrive/Facultad/3° año/Sistemas de Gestión de Bases de Datos/Compartido con Multipass/Primer laboratorio
❯ ls
test_db-master.zip

~/OneDrive/Facultad/3° año/Sistemas de Gestión de Bases de Datos/Compartido con Multipass/Primer laboratorio
❯ unzip test_db-master.zip
Archive:  test_db-master.zip
3c7fa05e04b4c339d91a43b7029a210212d48e6c
   creating: test_db-master/
  inflating: test_db-master/Changelog
  inflating: test_db-master/README.md
  inflating: test_db-master/employees.sql
  inflating: test_db-master/employees_partitioned.sql
  inflating: test_db-master/employees_partitioned_5.1.sql
   creating: test_db-master/images/
  inflating: test_db-master/images/employees.gif
  inflating: test_db-master/images/employees.jpg
  inflating: test_db-master/images/employees.png
  inflating: test_db-master/load_departments.dump
  inflating: test_db-master/load_dept_emp.dump
  inflating: test_db-master/load_dept_manager.dump
  inflating: test_db-master/load_employees.dump
  inflating: test_db-master/load_salaries1.dump
  inflating: test_db-master/load_salaries2.dump
  inflating: test_db-master/load_salaries3.dump
  inflating: test_db-master/load_titles.dump
  inflating: test_db-master/objects.sql
   creating: test_db-master/sakila/
  inflating: test_db-master/sakila/README.md
  inflating: test_db-master/sakila/sakila-mv-data.sql
  inflating: test_db-master/sakila/sakila-mv-schema.sql
  inflating: test_db-master/show_elapsed.sql
  inflating: test_db-master/sql_test.sh
  inflating: test_db-master/test_employees_md5.sql
  inflating: test_db-master/test_employees_sha.sql
  inflating: test_db-master/test_versions.sh

~/OneDrive/Facultad/3° año/Sistemas de Gestión de Bases de Datos/Compartido con Multipass/Primer laboratorio
❯
```

```
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio$ ls
test_db-master  test_db-master.zip
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio$ cd test_db-master
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio/test_db-master$ ls
Changelog                   employees.sql          load_dept_emp.dump      load_salaries1.dump  load_titles.dump  sakila           test_employees_md5.sql
employees_partitioned_5.1.sql  images              load_dept_manager.dump  load_salaries2.dump  objects.sql       show_elapsed.sql  test_employees_sha.sql
employees_partitioned.sql   load_departments.dump  load_employees.dump     load_salaries3.dump  README.md         sql_test.sh       test_versions.sh
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio/test_db-master$ mysql -u George\ Harrison -pmust_pass < employees.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:01:24
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio/test_db-master$
```

Para asegurarnos que se haya cargado correctamente la base de datos, podemos hacer la siguiente comprobación:



```
ubuntu@mv-sgbd ~/máquina local/Primer laboratorio/test_db-master$ mysql -u George\ Harrison -pmust_pass -t < test_employees_md5.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
+---------------------+
| INFO                |
+---------------------+
| TESTING INSTALLATION |
+---------------------+
+--------------+------------------+----------------------------------+
| table_name   | expected_records | expected_crc                     |
+--------------+------------------+----------------------------------+
| departments  |                9 | d1af5e170d2d1591d776d5638d71fc5f |
| dept_emp     |           331603 | ccf6fe516f990bdaa49713fc478701b7 |
| dept_manager |               24 | 8720e2f0853ac9096b689c14664f847e |
| employees    |           300024 | 4ec56ab5ba37218d187cf6ab09ce1aa1 |
| salaries     |          2844047 | fd220654e95aea1b169624ffe3fca934 |
| titles       |           443308 | bfa016c472df68e70a03facafa1bc0a8 |
+--------------+------------------+----------------------------------+
+--------------+------------------+----------------------------------+
| table_name   | found_records    | found_crc                        |
+--------------+------------------+----------------------------------+
| departments  |                9 | d1af5e170d2d1591d776d5638d71fc5f |
| dept_emp     |           331603 | ccf6fe516f990bdaa49713fc478701b7 |
| dept_manager |               24 | 8720e2f0853ac9096b689c14664f847e |
| employees    |           300024 | 4ec56ab5ba37218d187cf6ab09ce1aa1 |
| salaries     |          2844047 | fd220654e95aea1b169624ffe3fca934 |
| titles       |           443308 | bfa016c472df68e70a03facafa1bc0a8 |
+--------------+------------------+----------------------------------+
+--------------+---------------+-----------+
| table_name   | records_match | crc_match |
+--------------+---------------+-----------+
| departments  | OK            | ok        |
| dept_emp     | OK            | ok        |
| dept_manager | OK            | ok        |
| employees    | OK            | ok        |
| salaries     | OK            | ok        |
| titles       | OK            | ok        |
+--------------+---------------+-----------+
+------------------+
| computation_time |
+------------------+
| 00:01:13         |
+------------------+
+---------+--------+
| summary | result |
+---------+--------+
| CRC     | OK     |
| count   | OK     |
+---------+--------+
```

3) (25p) Para el siguiente ejercicio utilizar la base de datos “employees”. Se puede tomar como referencia el tutorial disponible en “Indices y Analizador Sintactico.pdf”. Considerar como apoyo el esquema de estructura de tablas disponible en la documentación de la base de datos “Employees”. Luego, para cada una de las consultas planteadas a continuación registrar:

- Cantidad de registros devueltos;
- Tiempos de ejecución;
- Información devuelta por la sentencia EXPLAIN (cantidad de filas, operaciones, índices, costos, etc.), ya sea en forma textual (en el formato de salida que consideren adecuado) o gráfica. Considerar el uso del modificador ANALYZE.

1. Listar los datos completos de todos los empleados con su respectivo historial de salarios;

```
mysql> SELECT * FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no;
| 499998 | 1956-09-05 | Patricia       | Breugel          | M      | 1993-10-13 | 499998 |  51182 | 2000-12-25 | 2001-12-25 |
| 499998 | 1956-09-05 | Patricia       | Breugel          | M      | 1993-10-13 | 499998 |  55003 | 2001-12-25 | 9999-01-01 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  63707 | 1997-11-30 | 1998-11-30 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  67043 | 1998-11-30 | 1999-11-30 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  70745 | 1999-11-30 | 2000-11-29 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  74327 | 2000-11-29 | 2001-11-29 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+----------------+------------------+--------+------------+--------+--------+------------+------------+
2844047 rows in set (4,78 sec)
```

```
mysql> EXPLAIN SELECT * FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no;
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref                | rows   | filtered | Extra |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL               | 299600 |   100.00 | NULL  |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      9 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
2 rows in set, 1 warning (0,00 sec)
```

```
mysql> EXPLAIN ANALYZE SELECT * FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=431970 rows=2.78e+6) (actual time=1.35..2990 rows=2.84e+6 loops=1)
    -> Table scan on e  (cost=30435 rows=299600) (actual time=1.15..205 rows=300024 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.412 rows=9.28) (actual time=0.00558..0.0081 rows=9.48 loops=300024)

1 row in set (3,76 sec)

mysql>
```

Puedo reunir estos comandos en un solo script:

```
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no;

EXPLAIN
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no;

EXPLAIN ANALYZE
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no\G
```

2. Al listado anterior agregar una condición (en WHERE) sobre el apellido (last_name), por ej. “LIKE ‘Gut%” o “= ‘González”’. Prever que la consulta devuelva más de un registro y esté ordenado por apellido.

Para este punto formulé el siguiente script:

```
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZÁLES'
ORDER BY e.last_name;

EXPLAIN
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZÁLES'
ORDER BY e.last_name;

EXPLAIN ANALYZE
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZÁLES'
ORDER BY e.last_name\G
```

Lo ejecuto con el siguiente comando dentro de la shell de MySQL:

```
mysql> SOURCE 3.2.sql
```

```
| 215104 | 1953-08-07 | Giap          | Guting    | F      | 1988-12-30 | 215104 |  67433 | 2000-03-09 | 2001-03-09 |
| 215104 | 1953-08-07 | Giap          | Guting    | F      | 1988-12-30 | 215104 |  68553 | 2001-03-09 | 2002-03-09 |
| 215104 | 1953-08-07 | Giap          | Guting    | F      | 1988-12-30 | 215104 |  70489 | 2002-03-09 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
1853 rows in set (0,22 sec)
```

```
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+----------------------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref               | rows   | filtered | Extra                      |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+----------------------------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL              | 299600 |    20.00 | Using where; Using filesort |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      9 |   100.00 | NULL                       |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+----------------------------+
2 rows in set, 1 warning (0,32 sec)
```

```
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=335351 rows=2.78e+6) (actual time=243..245 rows=1853 loops=1)
    -> Sort: e.last_name  (cost=30192 rows=299600) (actual time=243..243 rows=197 loops=1)
        -> Filter: ((e.last_name like 'GUT%') or (e.last_name = 'GONZÁLES'))  (cost=30192 rows=299600) (actual time=2.95..243 rows=197 loops=1)
            -> Table scan on e  (cost=30192 rows=299600) (actual time=0.76..174 rows=300024 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.451 rows=9.28) (actual time=0.00909..0.0117 rows=9.41 loops=197)

1 row in set (0,25 sec)
```

3. Al listado en el ejercicio 1. agregar una condición (en WHERE) para que el monto del salario (salary) cumpla algún criterio de igualdad o rango, por ejemplo “= 55555” o “BETWEEN 33333 AND 44444” o “> 70000 AND < 80000”. Prever que la consulta devuelva registros actuales (campos “to_date” = ‘9999-01-01’).

   El script que formulé para resolver este punto es el siguiente:

```sql
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01' AND s.salary > 7000;

EXPLAIN
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01' AND s.salary > 7000;

EXPLAIN ANALYZE
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01' AND s.salary > 7000\G
```

```
mysql> SOURCE 3.3.sql
```

```
| 499998 | 1956-09-05 | Patricia       | Breugel          | M      | 1993-10-13 | 499998 |  55003 | 2001-12-25 | 9999-01-01 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+----------------+------------------+--------+------------+--------+--------+------------+------------+
240124 rows in set (1,99 sec)

+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+---------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref               | rows    | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+---------+----------+-------------+
|  1 | SIMPLE      | s     | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL              | 2838426 |     3.33 | Using where |
|  1 | SIMPLE      | e     | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | employees.s.emp_no |       1 |   100.00 | NULL        |
+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+---------+----------+-------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=318484 rows=94605) (actual time=2.41..2213 rows=240124 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=285373 rows=94605) (actual time=2.39..1478 rows=240124 loops=1)
        -> Table scan on s  (cost=285373 rows=2.84e+6) (actual time=2.38..1137 rows=2.84e+6 loops=1)
    -> Single-row index lookup on e using PRIMARY (emp_no=s.emp_no)  (cost=0.25 rows=1) (actual time=0.00275..0.0028 rows=1 loops=240124)

1 row in set (2,29 sec)
```

4. Al listado en 1. agregarle conjuntamente las condiciones de 2. y 3. Esta consulta realizarla de manera distinta a las anteriores (p.e. distintos operadores, uso de CTE/TDL, etc.).

```
WITH empleadosFiltrados AS (
  SELECT *
  FROM employees AS e
  WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZALEZ'
)
SELECT *
FROM empleadosFiltrados AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.salary > 7000 AND s.to_date = '9999-01-01';

EXPLAIN
WITH empleadosFiltrados AS (
  SELECT *
  FROM employees AS e
  WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZALEZ'
)
SELECT *
FROM empleadosFiltrados AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.salary > 7000 AND s.to_date = '9999-01-01';

EXPLAIN ANALYZE
WITH empleadosFiltrados AS (
  SELECT *
  FROM employees AS e
  WHERE e.last_name LIKE 'GUT%' OR e.last_name = 'GONZALEZ'
)
SELECT *
FROM empleadosFiltrados AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no
WHERE s.salary > 7000 AND s.to_date = '9999-01-01'\G
```

```
| 495479 | 1956-01-17 | Odinaldo      | Guting    | F      | 1994-03-29 | 495479 | 101139 | 2002-03-27 | 9999-01-01 |
| 496790 | 1962-01-25 | Badri         | Guting    | M      | 1991-05-01 | 496790 |  78212 | 2002-04-28 | 9999-01-01 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  74449 | 2002-01-29 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
161 rows in set (0,20 sec)

+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref                | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL               | 299600 |    20.00 | Using where |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      9 |     3.33 | Using where |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=126040 rows=18541) (actual time=1.25..225 rows=161 loops=1)
    -> Filter: ((e.last_name like 'GUT%') or (e.last_name = 'GONZALEZ'))  (cost=30192 rows=59917) (actual time=1.22..222 rows=197 loops=1)
        -> Table scan on e  (cost=30192 rows=299600) (actual time=0.383..166 rows=300024 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=0.671 rows=0.309) (actual time=0.0113..0.012 rows=0.817 loops=197)
        -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.671 rows=9.28) (actual time=0.00796..0.0104 rows=9.41 loops=197)

1 row in set (0,23 sec)
```

5. Crear un índice por “last_name” en “employees” y ejecutar nuevamente las consultas de 1. a 4.

```
mysql> CREATE INDEX idx_last_name ON employees(last_name);
Query OK, 0 rows affected (2,14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Verificamos que se haya creado el índice correctamente:

```
mysql> SHOW INDEXES FROM employees\G
*************************** 1. row ***************************
        Table: employees
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: emp_no
    Collation: A
  Cardinality: 299600
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Visible: YES
   Expression: NULL
*************************** 2. row ***************************
        Table: employees
   Non_unique: 1
     Key_name: idx_last_name
 Seq_in_index: 1
  Column_name: last_name
    Collation: A
  Cardinality: 1585
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Visible: YES
   Expression: NULL
2 rows in set (0,00 sec)

mysql>
```

Ahora, ejecuto todos los scripts anteriores:

```
mysql> SOURCE 3.1.sql
```

```
| 499999 | 1958-05-01 | Sachin         | Tsukuda         | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+----------------+-----------------+--------+------------+--------+--------+------------+------------+
2844047 rows in set (4,17 sec)

+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref                | rows   | filtered | Extra |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL               | 299600 |   100.00 | NULL  |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      9 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=383913 rows=2.78e+6) (actual time=0.513..2657 rows=2.84e+6 loops=1)
    -> Table scan on e  (cost=30192 rows=299600) (actual time=0.488..172 rows=300024 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.252 rows=9.28) (actual time=0.00484..0.00717 rows=9.48 loops=300024)

1 row in set (3,43 sec)
```

```
mysql> SOURCE 3.2.sql
```

```
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  72654 | 2000-01-30 | 2001-01-29 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  73754 | 2001-01-29 | 2002-01-29 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  74449 | 2002-01-29 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
1853 rows in set (0,00 sec)

+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
| id | select_type | table | partitions | type  | possible_keys         | key           | key_len | ref                | rows | filtered | Extra                 |
+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
|  1 | SIMPLE      | e     | NULL       | range | PRIMARY,idx_last_name | idx_last_name | 66      | NULL               |  198 |   100.00 | Using index condition |
|  1 | SIMPLE      | s     | NULL       | ref   | PRIMARY               | PRIMARY       | 4       | employees.e.emp_no |    9 |   100.00 | NULL                  |
+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=323 rows=1838) (actual time=0.256..3.28 rows=1853 loops=1)
    -> Index range scan on e using idx_last_name over (last_name = 'GONZÁLES') OR ('GUT' <= last_name <= 'GUT????????????????????????????????????????????????????'), with index condition: ((e.last_name like 'GUT%') or (e.last_name = 'GONZÁLES'))  (cost=89.6 rows=198) (actual time=0.23..1.17 rows=197 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.257 rows=9.28) (actual time=0.00685..0.00951 rows=9.41 loops=197)

1 row in set, 1 warning (0,00 sec)
```

```
mysql> SOURCE 3.3.sql
```

```
| 499998 | 1956-09-05 | Patricia      | Breugel           | M      | 1993-10-13 | 499998 |  55003 | 2001-12-25 | 9999-01-01 |
| 499999 | 1958-05-01 | Sachin        | Tsukuda           | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+---------------+-------------------+--------+------------+--------+--------+------------+------------+
240124 rows in set (2,14 sec)

+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+---------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref                | rows    | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+---------+----------+-------------+
|  1 | SIMPLE      | s     | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL               | 2838426 |     3.33 | Using where |
|  1 | SIMPLE      | e     | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | employees.s.emp_no |       1 |   100.00 | NULL        |
+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+---------+----------+-------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=318484 rows=94605) (actual time=2.95..2220 rows=240124 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=285373 rows=94605) (actual time=2.92..1466 rows=240124 loops=1)
        -> Table scan on s  (cost=285373 rows=2.84e+6) (actual time=2.9..1133 rows=2.84e+6 loops=1)
    -> Single-row index lookup on e using PRIMARY (emp_no=s.emp_no)  (cost=0.25 rows=1) (actual time=0.00284..0.00288 rows=1 loops=240124)

1 row in set (2,30 sec)
```

```
mysql> SOURCE 3.4.sql
```

```
| 495479 | 1956-01-17 | Odinaldo      | Guting    | F      | 1994-03-29 | 495479 | 101139 | 2002-03-27 | 9999-01-01 |
| 496790 | 1962-01-25 | Badri         | Guting    | M      | 1991-05-01 | 496790 |  78212 | 2002-04-28 | 9999-01-01 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  74449 | 2002-01-29 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
161 rows in set (0,01 sec)

+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
| id | select_type | table | partitions | type  | possible_keys         | key           | key_len | ref                | rows | filtered | Extra                 |
+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
|  1 | SIMPLE      | e     | NULL       | range | PRIMARY,idx_last_name | idx_last_name | 66      | NULL               |  198 |   100.00 | Using index condition |
|  1 | SIMPLE      | s     | NULL       | ref   | PRIMARY               | PRIMARY       | 4       | employees.e.emp_no |    9 |     3.33 | Using where           |
+----+-------------+-------+------------+-------+-----------------------+---------------+--------+--------------------+------+----------+-----------------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=323 rows=61.3) (actual time=0.216..6.31 rows=161 loops=1)
    -> Index range scan on e using idx_last_name over (last_name = 'GONZALEZ') OR ('GUT' <= last_name <= 'GUT????????????????????????????????????????????????????'), with index condition: ((e.last_name like 'GUT%') or (e.last_name = 'GONZALEZ'))  (cost=89.6 rows=198) (actual time=0.186..1.67 rows=197 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=0.252 rows=0.309) (actual time=0.0211..0.0229 rows=0.817 loops=197)
        -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.252 rows=9.28) (actual time=0.014..0.0198 rows=9.41 loops=197)

1 row in set, 1 warning (0,01 sec)
```

6. Crear un índice por “salary” en “salaries” y ejecutar nuevamente las consultas de 1. a 4.

   Creo el índice:

```
mysql> CREATE INDEX idx_salary ON salaries(salary);
Query OK, 0 rows affected (10,44 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> SHOW INDEXES FROM salaries;
+----------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name   | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| salaries |          0 | PRIMARY    |            1 | emp_no      | A         |      305723 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| salaries |          0 | PRIMARY    |            2 | from_date   | A         |     2838426 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| salaries |          1 | idx_salary |            1 | salary      | A         |      105371 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0,00 sec)
```

```
mysql> SOURCE 3.1.sql
```

```
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  70745 | 1999-11-30 | 2000-11-29 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  74327 | 2000-11-29 | 2001-11-29 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+----------------+------------------+--------+------------+--------+--------+------------+------------+
2844047 rows in set (5,58 sec)

+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref                | rows   | filtered | Extra |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL               | 299600 |   100.00 | NULL  |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      9 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+---------+---------+--------------------+--------+----------+-------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=428109 rows=2.78e+6) (actual time=0.529..1694 rows=2.84e+6 loops=1)
    -> Table scan on e  (cost=30453 rows=299600) (actual time=0.467..115 rows=300024 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.399 rows=9.28) (actual time=0.00325..0.00465 rows=9.48 loops=300024)

1 row in set (2,10 sec)
```

```
mysql> SOURCE 3.2.sql
```

```
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  73754 | 2001-01-29 | 2002-01-29 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  74449 | 2002-01-29 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
1853 rows in set (0,01 sec)

+----+-------------+-------+------------+-------+-----------------------+-------------+---------+--------------------+------+----------+-----------------------+
| id | select_type | table | partitions | type  | possible_keys         | key         | key_len | ref                | rows | filtered | Extra                 |
+----+-------------+-------+------------+-------+-----------------------+-------------+---------+--------------------+------+----------+-----------------------+
|  1 | SIMPLE      | e     | NULL       | range | PRIMARY,idx_last_name | idx_last_name | 66    | NULL               |  198 |   100.00 | Using index condition |
|  1 | SIMPLE      | s     | NULL       | ref   | PRIMARY               | PRIMARY     | 4       | employees.e.emp_no |    9 |   100.00 | NULL                  |
+----+-------------+-------+------------+-------+-----------------------+-------------+---------+--------------------+------+----------+-----------------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=353 rows=1838) (actual time=0.33..4.27 rows=1853 loops=1)
    -> Index range scan on e using idx_last_name over (last_name = 'GONZÁLES') OR ('GUT' <= last_name <= 'GUT??????????????????????????????????????????????????????'), with index condition: ((e.last_name like 'GUT%') or (e.last_name = 'GONZÁLES'))  (cost=108 rows=198) (actual time=0.303..1.32 rows=197 loops=1)
    -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.31 rows=9.28) (actual time=0.00998..0.0135 rows=9.41 loops=197)

1 row in set, 1 warning (0,01 sec)
```

```
mysql> SOURCE 3.3.sql
```

```
| 499998 | 1956-09-05 | Patricia       | Breugel          | M      | 1993-10-13 | 499998 |  55003 | 2001-12-25 | 9999-01-01 |
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 |
+--------+------------+----------------+------------------+--------+------------+--------+--------+------------+------------+
240124 rows in set (2,53 sec)

+----+-------------+-------+------------+--------+--------------------+---------+---------+--------------------+---------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys      | key     | key_len | ref                | rows    | filtered | Extra       |
+----+-------------+-------+------------+--------+--------------------+---------+---------+--------------------+---------+----------+-------------+
|  1 | SIMPLE      | s     | NULL       | ALL    | PRIMARY,idx_salary | NULL    | NULL    | NULL               | 2838426 |     5.00 | Using where |
|  1 | SIMPLE      | e     | NULL       | eq_ref | PRIMARY            | PRIMARY | 4       | employees.s.emp_no |       1 |   100.00 | NULL        |
+----+-------------+-------+------------+--------+--------------------+---------+---------+--------------------+---------+----------+-------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=344168 rows=141921) (actual time=3.83..2591 rows=240124 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=285605 rows=141921) (actual time=3.73..1772 rows=240124 loops=1)
        -> Table scan on s  (cost=285605 rows=2.84e+6) (actual time=3.72..1439 rows=2.84e+6 loops=1)
    -> Single-row index lookup on e using PRIMARY (emp_no=s.emp_no)  (cost=0.313 rows=1) (actual time=0.0031..0.00314 rows=1 loops=240124)

1 row in set (2,67 sec)
```

```
mysql> SOURCE 3.4.sql
```

```
| 495479 | 1956-01-17 | Odinaldo      | Guting    | F      | 1994-03-29 | 495479 | 101139 | 2002-03-27 | 9999-01-01 |
| 496790 | 1962-01-25 | Badri         | Guting    | M      | 1991-05-01 | 496790 |  78212 | 2002-04-28 | 9999-01-01 |
| 497316 | 1955-01-06 | Gonzalo       | Guting    | M      | 1986-03-31 | 497316 |  74449 | 2002-01-29 | 9999-01-01 |
+--------+------------+---------------+-----------+--------+------------+--------+--------+------------+------------+
161 rows in set (0,01 sec)

+----+-------------+-------+------------+------+-----------------------+--------------+---------+-------------------+------+----------+-----------------------+
| id | select_type | table | partitions | type | possible_keys         | key          | key_len | ref               | rows | filtered | Extra                 |
+----+-------------+-------+------------+------+-----------------------+--------------+---------+-------------------+------+----------+-----------------------+
|  1 | SIMPLE      | e     | NULL       | range | PRIMARY,idx_last_name | idx_last_name | 66      | NULL              |  198 |   100.00 | Using index condition |
|  1 | SIMPLE      | s     | NULL       | ref  | PRIMARY,idx_salary    | PRIMARY      | 4       | employees.e.emp_no |    9 |     5.00 | Using where           |
+----+-------------+-------+------------+------+-----------------------+--------------+---------+-------------------+------+----------+-----------------------+
2 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (cost=344 rows=91.9) (actual time=0.206..6.36 rows=161 loops=1)
    -> Index range scan on e using idx_last_name over (last_name = 'GONZALEZ') OR ('GUT' <= last_name <= 'GUT????????????????????????????????????????????????????'), with index condition: ((e.last_name like 'GUT%')
 or (e.last_name = 'GONZALEZ'))  (cost=102 rows=198) (actual time=0.179..1.32 rows=197 loops=1)
    -> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary > 7000))  (cost=0.291 rows=0.464) (actual time=0.0244..0.0251 rows=0.817 loops=197)
        -> Index lookup on s using PRIMARY (emp_no=e.emp_no)  (cost=0.291 rows=9.28) (actual time=0.0205..0.0233 rows=9.41 loops=197)

1 row in set, 1 warning (0,01 sec)
```

7. Teniendo en cuenta que hay distintos cargos de Ingenieros (Engineer, Senior Engineer y Assistant Engineer en la tabla titles), listar todos los datos de los empleados con el historial de salarios y cargos ocupados, pero sólo para los cargos de Ingenieros. Hacer esta consulta evaluando el campo title una vez con “LIKE” (para considerar todos los cargos de una vez) y otra con “=” (considerando los distintos cargos por separado).

Script con cláusula **LIKE**:

```sql
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title LIKE '%Engineer%'
ORDER BY e.emp_no, t.from_date;

EXPLAIN
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title LIKE '%Engineer%'
ORDER BY e.emp_no, t.from_date;

EXPLAIN ANALYZE
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title LIKE '%Engineer%'
ORDER BY e.emp_no, t.from_date\G
```

```
mysql> SOURCE 3.7.like.sql
```

```
| 499999 | 1958-05-01 | Sachin        | Tsukuda          | M      | 1997-11-30 | 499999 |  74327 | 2000-11-29 | 2001-11-29 | 499999 | Engineer          | 1997-11-30 | 9999-01-01 |
| 499999 | 1958-05-01 | Sachin        | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 | 499999 | Engineer          | 1997-11-30 | 9999-01-01 |
+--------+------------+---------------+------------------+--------+------------+--------+--------+------------+------------+--------+-------------------+------------+------------+
2396159 rows in set (11,66 sec)

+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+--------+----------+----------------------------------------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref                | rows   | filtered | Extra                                        |
+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+--------+----------+----------------------------------------------+
|  1 | SIMPLE      | t     | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL               | 441237 |    11.11 | Using where; Using temporary; Using filesort |
|  1 | SIMPLE      | e     | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | employees.t.emp_no |      1 |   100.00 | NULL                                         |
|  1 | SIMPLE      | s     | NULL       | ref    | PRIMARY       | PRIMARY | 4       | employees.t.emp_no |      9 |   100.00 | NULL                                         |
+----+-------------+-------+------------+--------+---------------+---------+---------+--------------------+--------+----------+----------------------------------------------+
3 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Sort: e.emp_no, t.from_date  (actual time=10040..10577 rows=2.4e+6 loops=1)
    -> Stream results  (cost=129958 rows=455130) (actual time=0.95..7356 rows=2.4e+6 loops=1)
        -> Nested loop inner join  (cost=129958 rows=455130) (actual time=0.932..3924 rows=2.4e+6 loops=1)
            -> Nested loop inner join  (cost=66912 rows=49021) (actual time=0.774..1166 rows=227881 loops=1)
                -> Filter: (t.title like '%Engineer%')  (cost=44568 rows=49021) (actual time=0.746..510 rows=227881 loops=1)
                    -> Table scan on t  (cost=44568 rows=441237) (actual time=0.74..335 rows=443308 loops=1)
                -> Single-row index lookup on e using PRIMARY (emp_no=t.emp_no)  (cost=0.356 rows=1) (actual time=0.00254..0.00259 rows=1 loops=227881)
            -> Index lookup on s using PRIMARY (emp_no=t.emp_no)  (cost=0.358 rows=9.28) (actual time=0.0074..0.0107 rows=10.5 loops=227881)

1 row in set (11,45 sec)

mysql>
```

Script con operador **=**:

```
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title = 'Engineer' OR t.title = 'Senior Engineer' OR t.title = 'Assistant Engineer'
ORDER BY e.emp_no, t.from_date;

EXPLAIN
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title = 'Engineer' OR t.title = 'Senior Engineer' OR t.title = 'Assistant Engineer'
ORDER BY e.emp_no, t.from_date;

EXPLAIN ANALYZE
SELECT *
FROM employees AS e INNER JOIN salaries AS s ON e.emp_no = s.emp_no INNER JOIN titles AS t ON e.emp_no = t.emp_no
WHERE t.title = 'Engineer' OR t.title = 'Senior Engineer' OR t.title = 'Assistant Engineer'
ORDER BY e.emp_no, t.from_date\G
```

```
mysql> SOURCE 3.7.equal.sql
```

```
| 499999 | 1958-05-01 | Sachin         | Tsukuda          | M      | 1997-11-30 | 499999 |  77303 | 2001-11-29 | 9999-01-01 | 499999 | Engineer           | 1997-11-30 | 9999-01-01 |
+--------+------------+----------------+------------------+--------+------------+--------+--------+------------+------------+--------+--------------------+------------+------------+
2396159 rows in set (12,56 sec)

+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------+--------+----------+----------------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref               | rows   | filtered | Extra                                        |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------+--------+----------+----------------------------------------------+
|  1 | SIMPLE      | t     | NULL       | ALL   | PRIMARY       | NULL    | NULL    | NULL              | 441237 |    27.10 | Using where; Using temporary; Using filesort |
|  1 | SIMPLE      | e     | NULL       | eq_ref | PRIMARY      | PRIMARY | 4       | employees.t.emp_no |      1 |   100.00 | NULL                                         |
|  1 | SIMPLE      | s     | NULL       | ref   | PRIMARY       | PRIMARY | 4       | employees.t.emp_no |      9 |   100.00 | NULL                                         |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------+--------+----------+----------------------------------------------+
3 rows in set, 1 warning (0,00 sec)

*************************** 1. row ***************************
EXPLAIN: -> Sort: e.emp_no, t.from_date  (actual time=9527..10091 rows=2.4e+6 loops=1)
    -> Stream results  (cost=253077 rows=1.11e+6) (actual time=1.1..6980 rows=2.4e+6 loops=1)
        -> Nested loop inner join  (cost=253077 rows=1.11e+6) (actual time=1.03..3767 rows=2.4e+6 loops=1)
            -> Nested loop inner join  (cost=99174 rows=119575) (actual time=0.761..1109 rows=227881 loops=1)
                -> Filter: ((t.title = 'Engineer') or (t.title = 'Senior Engineer') or (t.title = 'Assistant Engineer'))  (cost=44569 rows=119575) (actual time=0.727..487 rows=227881 loops=1)
                    -> Table scan on t  (cost=44569 rows=441237) (actual time=0.72..317 rows=443308 loops=1)
                -> Single-row index lookup on e using PRIMARY (emp_no=t.emp_no)  (cost=0.357 rows=1) (actual time=0.0024..0.00245 rows=1 loops=227881)
            -> Index lookup on s using PRIMARY (emp_no=t.emp_no)  (cost=0.359 rows=9.28) (actual time=0.00717..0.0104 rows=10.5 loops=227881)

1 row in set (10,99 sec)

mysql>
```

8. Escribir un breve resumen comparativo (no más de 15 renglones) como conclusión de los datos registrados en cada ejecución.

El uso de índices en las tablas puede mejorar significativamente el rendimiento de las consultas, especialmente aquellas que incluyen cláusulas `WHERE`, `JOIN`, `ORDER BY` o `GROUP BY`. Sin embargo, las consultas `SELECT *` no siempre se benefician de los índices y, en algunos casos, pueden volverse más lentas, ya que el motor de la base de datos podría estar gestionando los índices innecesariamente.

Es importante mantener los índices actualizados, ya que su actualización afecta principalmente a las operaciones de escritura (`INSERT`, `UPDATE`, `DELETE`), pero también puede influir en las consultas de lectura. Los índices ocupan espacio en disco y memoria, y su gestión constante puede generar una sobrecarga si no se utilizan adecuadamente en las consultas.

Cuando se ejecuta un `SELECT *`, el sistema podría estar calculando y gestionando índices que no son necesarios, lo que podría incrementar el tiempo de respuesta. Por lo tanto, aunque los índices son esenciales para optimizar las consultas, es crucial diseñarlos cuidadosamente y evaluar su impacto en el rendimiento general del sistema.